

Toggle

Toggle A sleek and intuitive switch component for toggling between binary states (on/off, enabled/disabled). Features smooth animations, multiple sizes, flexible label positioning, and comprehensive accessibility support for enhanced user experience.

How to use ■ `import { AavaToggleComponent } from "@aava/play-core"` ; Basic Usage ■ Simple toggle implementations with labels and basic functionality.

Sizes ■ Four size variants to accommodate different interface requirements and visual hierarchy.

Available Sizes ■ `xs` (Extra Small) - Extra compact size for very dense interfaces `sm` (Small) : Compact for dense interfaces and secondary controls `md` (Medium) : Standard size for most use cases (default) `lg` (Large) : Prominent for primary settings and better accessibility

Positions ■ Flexible label positioning for optimal layout integration.

Position Options ■ `Left` : Label appears to the left of the switch (default) `Right` : Label appears to the right of the switch

States ■ Different toggle states including checked, unchecked, and disabled variations.

State Management ■ `Checked` : Active/enabled state with visual feedback `Unchecked` : Inactive/disabled state `Disabled` : Non-interactive state with reduced opacity

Focus : Keyboard navigation with clear focus indicators

Animation ■ Events ■ Event handling for user interactions and state changes.

Event Features ■ `State Change` : Emits boolean value on toggle

Keyboard Support : Space and Enter key activation

Click Handling : Mouse and touch interaction support

Disabled Prevention : No events when disabled

Thumb Icon ■ Add icons to the toggle thumb to visually indicate checked and unchecked states. This is useful for enhancing clarity, especially in settings or preference toggles.

Features ■ Show different icons for checked and unchecked states

Fully customizable icon names (uses Lucide icon set)

Works with all sizes and positions

Accessible and keyboard-friendly

Accessibility ■ Built-in accessibility features ensuring inclusive user experience.

Accessibility Features ■ `Keyboard Navigation` : Tab navigation and Space/Enter activation

ARIA Support : Proper switch role and state announcements

Screen Reader : Descriptive labels and state information

Focus Management : Clear visual focus indicators

High Contrast : Enhanced borders and outlines for visibility

Motion Preferences : Respects reduced motion accessibility settings

API Reference ■ Inputs ■

Property Type	Default	Description
<code>size</code>	<code>'xs'</code>	Size variant of the toggle switch
<code>title</code>	<code>string</code>	Label text displayed next to the toggle
<code>position</code>	<code>'left'</code>	Position of the label relative to switch
<code>disabled</code>	<code>boolean false</code>	Whether the toggle is disabled
<code>checked</code>	<code>boolean false</code>	Current checked state of the toggle
<code>animation</code>	<code>boolean true</code>	Whether to enable smooth animations
<code>showIcons</code>	<code>boolean false</code>	Show icons in the thumb for checked/unchecked
<code>checkedIcon</code>	<code>string 'check'</code>	Icon name for the checked state (Lucide icon)
<code>uncheckedIcon</code>	<code>string 'x'</code>	Icon name for the unchecked state (Lucide icon)

Outputs ■

Event Type	Description
<code>checkedChange</code>	<code>EventEmitter<boolean></code> Emitted when toggle state changes

Methods ■

Method	Parameters	Return Type	Description
<code>onToggle()</code>	-	<code>void</code>	Toggle the checked state (if not disabled)
<code>onKeyDown()</code>	<code>event: KeyboardEvent</code>	<code>void</code>	Handle keyboard activation (Space/Enter)

Properties ■

Property Type	Description
<code>titleName</code>	<code>string null</code> Generated ID for label association

CSS Custom Properties ■

Property	Description
<code>toggle-checked</code>	Property Description

--toggle-icon-checked-color Color of the icon when toggle is checked
--toggle-icon-unchecked-color Color of the icon when toggle is unchecked --text-color-secondary
Secondary text color used in toggle --toggle-gap Gap between toggle and label --toggle-label-text
Color of the label text --toggle-font-size-xs Font size for extra small label --toggle-font-size-sm
Font size for small label --toggle-font-size-md Font size for medium label --toggle-font-size-lg Font
size for large label --toggle-label-disabled-text Color of label text when toggle is disabled
--toggle-border-radius Border radius of toggle elements --toggle-track-background Background
color of the toggle track --toggle-track-checked-background-gradient Background gradient for
checked state --toggle-size-xs-width Width of extra small toggle switch --toggle-size-xs-height
Height of extra small toggle switch --toggle-size-xs-thumb-size Size of the thumb for extra small
toggle --toggle-size-sm-width Width of small toggle switch --toggle-size-sm-height Height of small
toggle switch --toggle-size-sm-thumb-size Size of the thumb for small toggle
--toggle-size-md-width Width of medium toggle switch --toggle-size-md-height Height of medium
toggle switch --toggle-size-md-thumb-size Size of the thumb for medium toggle
--toggle-size-lg-width Width of large toggle switch --toggle-size-lg-height Height of large toggle
switch --toggle-size-lg-thumb-size Size of the thumb for large toggle --toggle-glass-shadow Glass
shadow effect for toggle --toggle-thumb-background Background color of the toggle thumb

Best Practices

- **Design Guidelines**
 - Use toggles for immediate binary actions that take effect instantly
 - Place labels on the left for left-to-right reading patterns
 - Choose appropriate sizes based on interface hierarchy
 - Use disabled state for features unavailable to current user
 - Consider animation preferences for better accessibility
- **Accessibility**
 - Always provide meaningful titles that describe the toggle purpose
 - Use proper ARIA attributes for screen reader compatibility
 - Ensure sufficient color contrast for all states
 - Test keyboard navigation thoroughly
 - Consider reduced motion preferences
- **Performance**
 - Avoid frequent programmatic state changes that trigger animations
 - Use OnPush change detection strategy for optimal performance
 - Consider disabling animations in performance-critical scenarios

```

<aava-toggle
  size="md"
  title="Enable Notifications"
  position="left"
  [checked]="notificationsEnabled"
  (checkedChange)="onNotificationsChange($event)"
>
</aava-toggle>

```

```

notificationsEnabled = false;

onNotificationsChange(checked: boolean) {
  this.notificationsEnabled = checked;
  console.log('Notifications:', checked ? 'enabled' : 'disabled');
}

```

```

<aava-toggle size="xs" title="Extra Small"></aava-toggle>

```

```

<aava-toggle size="sm" title="Small"></aava-toggle>

```

```

<aava-toggle size="md" title="Medium"></aava-toggle>

```

```

<aava-toggle size="lg" title="Large"></aava-toggle>

```

```

<aava-toggle size="md" title="Left Position" position="left" [checked]="true">
</aava-toggle>

```

```

<aava-toggle
  size="md"
  title="Right Position"
  position="right"
  [checked]="false"
>
</aava-toggle>

```

```
<aava-toggle size="md" title="Enabled Unchecked" [checked]="false">
</aava-toggle>
```

```
<aava-toggle size="md" title="Enabled Checked" [checked]="true"> </aava-toggle>
```

```
<aava-toggle
  size="md"
  title="Disabled Unchecked"
  [disabled]="true"
  [checked]="false"
>
</aava-toggle>
```

```
<aava-toggle
  size="md"
  title="Disabled Checked"
  [disabled]="true"
  [checked]="true"
>
</aava-toggle>
```

```
<!-- With Animation -->
<aava-toggle size="md" title="Animated Toggle" [animation]="true">
</aava-toggle>
```

```
<!-- Without Animation -->
<aava-toggle size="md" title="Non-Animated Toggle" [animation]="false">
</aava-toggle>
```

```

<aava-toggle
  size="md"
  title="Event Toggle"
  position="left"
  [checked]="eventToggleEnabled"
  (checkedChange)="onEventToggleChange($event)"
>
</aava-toggle>

```

```

eventToggleEnabled = false;

onEventToggleChange(checked: boolean) {
  this.eventToggleEnabled = checked;
  const timestamp = new Date().toLocaleTimeString();
  this.eventLogs.unshift(
    `[${timestamp}] Event toggle changed to: ${checked ? 'enabled' : 'disabled'}`
  );

  // Keep only last 10 entries
  if (this.eventLogs.length > 10) {
    this.eventLogs = this.eventLogs.slice(0, 10);
  }

  console.log('Event toggle:', checked ? 'enabled' : 'disabled');
}

```

■ No code found

■ No code found